IN THE CLAIMS:

Please find below a listing of all of the pending claims. The statuses of the claims are set forth in parentheses.

1. (Currently Amended) A monitoring method for a component-based software system operating over one or more processing devices, comprising the steps of:

initiating an invocation of a second software component from within an execution of a first software component;

recording a stub start log data including a global causal identifier in an instrumented stub before said invocation of said second software component;

recording a stub end log data including the global causal identifier in said instrumented stub after a response is received from said invocation of said second software component, said response including the global causal identifier;

wherein said stub start log data and said stub end log data gather runtime information about execution of said second software component within said component-based software system.

2. (Original) The method of claim 1, wherein said instrumented stub is generated from a description of an interface of said second software component.

3. (Original) The method of claim 1, wherein said second software component is remote from said first software component.

4. (Original) The method of claim 1, wherein said first software component resides on a first processing device and said second software component resides on a second processing device.

5. (Original) The method of claim 1, further comprising the preliminary step of selecting a log data contents to be included in said stub start and stub end log data, with the selecting step logging zero or more of an application semantic behavior data, a timing latency data, a shared resource usage data, and a causality relationship data.

6. (Original) The method of claim 1, wherein a log data contents is configured during generation of said instrumented stub.

7. (Original) The method of claim 1, wherein a log data contents is configured during operation of said component-based software system.

8. (Original) The method of claim 7, wherein a runtime information generated during said operation of said component-based software system includes a regular expression that determines a particular log data contents, and wherein a user is capable of changing said particular log data contents during said operation of said component-based software system by setting said regular expression.

9. (Original) The method of claim 1, further comprising the steps of:

initiating said invocation of said second software component from within an execution of an instrumented skeleton;

recording a skeleton start log data before said instrumented skeleton invokes said second software component; and

recording a skeleton end log data in said instrumented skeleton after a response is received from said invocation of said second software component.

10. (Original) The method of claim 9, wherein said instrumented skeleton is generated from a description of an interface of said second software component.

11. (Original) The method of claim 9, wherein said instrumented skeleton is generated from a description of an interface of said second software component and wherein said second software component is remote from said first software component.

12. (Original) The method of claim 9, wherein a particular instrumented stub is capable of enabling and disabling a data logging capability of a corresponding instrumented skeleton.

13. (Original) The method of claim 9, wherein an accumulated log data from a plurality of instrumented stubs and a plurality of instrumented skeletons is collected and correlated.

14. (Original) The method of claim 9, wherein said stub start, stub end, skeleton start, and skeleton end log data capture a causality relationship data between said first software component and said second software component.

15. (Original) The method of claim 9, wherein said stub start, stub end, skeleton start, and skeleton end log data are used to determine a causality relationship data for a plurality of threads.

16. (Original) The method of claim 9, wherein said stub start, stub end, skeleton start, and skeleton end log data are used to determine a causality relationship data for a plurality of threads spawned during invocation of said second software component.

17. (Original) The method of claim 9, wherein said stub start, stub end, skeleton start, and skeleton end log data are used to determine a causality relationship data for a thread in which said first software component is invoked.

18. (Original) The method of claim 9, further comprising the preliminary step of selecting a log data contents to be included in said skeleton start and skeleton end log data, with the selecting step logging zero or more of a timing latency data, a shared resource usage data, and a causality relationship data.

19. (Original) The method of claim 9, wherein the method includes a transportation of at least a portion of said stub start log data of said instrumented stub to said instrumented skeleton.

20. (Currently Amended) The method of claim 19, wherein said transportation is accomplished by passing the global causal identifier ~~an additional private parameter~~ to a function defined in an interface definition of said second software component.

21. (Original) The method of claim 9, wherein said instrumented skeleton stores at least a portion of said skeleton start log data to a thread-specific storage.

22. (Original) The method of claim 21, wherein an event number included in said at least a portion of said skeleton start log data is updated before being copied into said thread-specific storage.

23. (Original) The method of claim 9, further comprising the steps of:

retrieving a thread-transportable log data from a thread-specific storage of a parent thread;

transporting said thread-transportable log data to a child thread;

adding a thread information about a child thread to said thread-transportable log data to form a child thread data; and

recording said child thread data to a thread table of said child thread.

24. (Original) The method of claim 23, wherein said thread-transportable log data comprises a self thread identifier and optionally a function container identifier, with said self thread identifier distinguishing user-application generated threads from threads generated by an underlying component-based system runtime infrastructure.

25. (Original) The method of claim 9, further comprising the step of intercepting dynamic memory allocation and de-allocation requests and logging a heap memory usage data from said requests.

26. (Original) The method of claim 9, wherein a particular log data is recorded in a per- process log table.

27. (Original) The method of claim 9, wherein a particular log data is recorded on a per-thread basis.

28. (Original) The method of claim 9, wherein a particular log data is stored in a persistent storage.

29. (Currently Amended) A monitoring method for a component-based software system operating over one or more processing devices, comprising the steps of:

accumulating one or more stub start log data entries <u>including a global causal identifier</u>, with a stub start log data entry of said one or more stub start data entries being recorded by an instrumented stub before a subsequent software component invocation;

accumulating one or more skeleton start log data entries including the global causal identifier, with a skeleton start log data entry of said one or more skeleton start data entries being recorded by an instrumented skeleton before said instrumented skeleton invokes said subsequent software component;

accumulating one or more skeleton end log data entries including the global causal identifier, with a skeleton end log data entry of said one or more skeleton end log data entries being recorded by said instrumented skeleton after a response is received from said subsequent software component invocation;

accumulating one or more stub end log data entries including the global causal identifier, with a stub end log data entry of said one or more stub end log data entries being recorded by said instrumented stub after said response is received from said subsequent software component invocation; and

processing an accumulated log data, using the global causal identifier, and calculating a system behavior characteristic for one or more software components executing within said component-based software system.

30. (Original) The method of claim 29, wherein said system behavior characteristic comprises a causality relationship data.

31. (Original) The method of claim 29, wherein said system behavior characteristic comprises an application semantic behavior data.

32. (Original) The method of claim 29, wherein said system behavior characteristic comprises a shared resource usage data.

33. (Original) The method of claim 29, wherein said system behavior characteristic comprises a shared resource usage data, with said shared resource usage data including a CPU usage data.

34. (Original) The method of claim 29, wherein said system behavior characteristic comprises a shared resource usage data, with said shared resource usage data including a memory usage data.

35. (Original) The method of claim 29, wherein said system behavior characteristic comprises a timing latency data.

36. (Currently Amended) A computer system adapted to monitor component-based software applications, comprising:

at least one processing device residing in said computer system; one or more software components residing on said at least one processing device and capable of executing in said computer system; and

one or more instrumented stubs in said one or more software components, with an instrumented stub being capable of recording a stub start log data at an execution invocation of said instrumented stub in a first software component and recording a stub end log data at an execution conclusion of said instrumented stub, said stub start log data and said stub end

log data including a global causal identifier and wherein said one or more instrumented stubs is configured to transmit said global causal identifier to at least one other component in the computer system.

37. (Currently Amended) The system of claim 36, further comprising a memory capable of storing said ~~first~~ stub start and stub end log data.

38. (Original) The system of claim 36, further comprising one or more instrumented skeletons, with an instrumented skeleton being capable of recording a skeleton start log data at an execution invocation of said instrumented skeleton in a second software component and recording a skeleton end log data at an execution conclusion of said instrumented skeleton.

39. (Original) The system of claim 36, wherein a first software component of said one or more software components is capable of invoking a second software component.

40. (Original) The system of claim 36, wherein a first software component of said one or more software components is capable of invoking a second software component and wherein said first software component resides on a first processing device and said second software component resides on a second processing device.

41. (Original) The system of claim 36, wherein said memory further includes a thread table adapted to store thread log data.

42. (Original) The system of claim 36, wherein said component-based software system further comprises a persistent storage capable of collecting a plurality of log data.

43. (Original) The system of claim 36, further comprising:

a persistent storage capable of collecting a plurality of log data;

an analyzer communicating with said persistent storage and capable of retrieving and analyzing log data from said persistent storage; and

a monitoring coordinator communicating with one or more instrumented, component-based software applications and capable of enabling or disabling instrumented stubs and instrumented skeletons.